

Feed-Forward Staircase Codes

Lei M. Zhang and Laurent Schmalen

Abstract—We propose two variants of staircase codes that resolve the issue of parity-propagation in their encoding process. The proposed codes provide a systematic way of terminating a staircase code after an arbitrary number of blocks. The class of feed-forward staircase codes are introduced, which uses a self-protection technique to avoid parity-propagation. We also introduce the class of partial feed-forward staircase codes, which allows parity-propagation to occur over a given number of blocks. By amortizing the complexity of self-protection over several standard staircase blocks, the encoding complexity of these codes is made comparable to staircase codes. Partial feed-forward staircase codes have the same error-floor as staircase codes. Simulations confirm that the performance of the proposed codes in both the waterfall and error-floor regions is similar to the original staircase codes. The proposed codes help extend the domain of application of staircase codes to systems in which parity-propagation is undesirable or termination is necessary.

I. INTRODUCTION

HIGH-SPEED fiber optical communication system are a challenging environment for forward error correction (FEC) schemes. Modern high-speed optical communication systems require high-performing FEC engines that support throughputs of 100 Gbit/s and multiples thereof, that have low power consumption, that realize net coding gains (NCGs) close to the theoretical capacity limits at a target BER of 10^{-15} , and that are preferably adapted to the peculiarities of the optical channel [1].

Although coding schemes that allow for soft-decision decoding are now well established in optical communications [1], especially in long-haul and submarine transmission systems which need to operate at the lowest possible signal-to-noise ratio (SNR), hard-decision decoding is still predominant in the widely deployed metro networks, due to its low complexity leading to power-friendly receiver implementations [2]. Such low-complexity receivers are also attractive for data center interconnect applications.

In the recent years, several new capacity-approaching coding schemes suitable for high-speed optical communications have been presented. Staircase codes [3], [4], are hard-decision decoded, spatially-coupled codes with practical application in forward error-correction for long-haul optical-fiber transmission systems. An ITU-T G.709-compatible staircase code with rate $R = 239/255$ was shown to operate within 0.56 dB of the

capacity of the binary-input AWGN channel with hard decision at the output (which is equivalent to a binary symmetric channel (BSC)) at a bit-error rate (BER) of 10^{-15} [3]. Its gap to capacity is smaller than all of the enhanced coding schemes proposed in ITU-T recommendation G.975.1 [5]. In [4], staircase codes with rates $R \geq 6/7$ were shown to be within 0.80 dB of capacity in terms of NCG at a BER of 10^{-15} . Such coding gains are obtained by using an iterative, hard-decision decoding algorithm with decoder data-flow orders of magnitude lower than that of message-passing decoders for sparse-graph codes such as Turbo or Low-Density Parity-Check (LDPC) codes [3]. For long-haul optical-fiber transmissions systems where bit-rates exceed 100 Gb/s, staircase codes are often the best practical solution.

Besides staircase code and variants thereof [6], several other code constructions based on spatial coupling of algebraic component codes have been proposed, e.g., braided BCH codes [7]. Recently, multiple works show that these codes can approach capacity of the BSC under simple iterative decoding when the rate is large enough [8]–[11]. However, all the proposed structures of spatially coupled algebraic product codes are recursive codes which lead to several practical drawbacks in their implementation: First, a recursive structure requires extra circuitry [12] for terminating the code, which may be undesired in some applications where a low-complexity decoder implementation is crucial. Previous publications have not explicitly dealt with code termination but have only considered free-running, non-terminated codes. Terminating a feed-forward code on the other hand is straightforward.

A second drawback of recursive codes is the effect of *parity-propagation*; a single non-zero information bit leads to an infinitely extending parity sequence. This effect may be undesired in some optical transmission applications, where the transceivers are usually free-running due to the setup times of links [13] and only some of the transmitted bits carry useful information. Parity propagation limits in this case the possibility of switching off the forward error correction circuitry during times when no useful data is transmitted, non-negligibly increasing the transceiver power consumption [2].

In this paper, we discuss several options for constructing feed-forward staircase codes. It becomes quickly obvious that a straightforward modification of the staircase encoding structure to avoid parity propagation will lead to unacceptably high error floors for most applications. In order to mitigate the error floor, we use the technique of self-protecting parity-bits [5, App. I.9] together with a clever interleaving to construct a class of feed-forward staircase codes. We also give an approximation of the expected error floor based on the minimum size stall pattern. In some applications with very stringent requirements, the error floor may still be too large. For this reason, in the second part of the paper, we slightly

The associate editor coordinating the review of this letter and approving it for publication was Dr. xxx. Manuscript received XXX. yy, 2016.

L. M. Zhang is with the University of Toronto, ECE department. His work has been carried out while he was visiting Nokia Bell Labs funded by a scholarship from the German DAAD-RisePro program.

L. Schmalen is with Nokia Bell Labs, Stuttgart, Germany (e-mail: first.last@nokia.com).

L. Schmalen was supported by the German BMBF in the scope of the CELTIC+ project SENDATE-TANDEM.

Digital Object Identifier xx.xxxx/xxx.2016.xxxxxx

relax the parity-propagation constraint and present *partial feed-forward* staircase codes, which have the same error floor as the original staircase codes but completely avoid parity-propagation and allow for easy termination.

This paper is structured as follows: In Sec. II, we introduce the basic notation and recapitulate the structure and main properties of staircase codes. In Sec. III, we introduce a first construction of feed-forward staircase codes based on self-protected parity-bits. In Sec. IV, we slightly generalize this construction and introduce partial feed-forward staircase codes, which have a slightly lower rate but improved error floor properties. Error floor approximations based on minimal stall patterns are derived in Sec. V. Finally, we compare in Sec. VI the performance of both schemes using a coding setup typically used in optical communications.

II. BACKGROUND: STAIRCASE CODES

In this section, we briefly overview the encoding and decoding of staircase codes since the proposed codes share many common features with the original staircase code. We also motivate our work by examining the parity-propagation property of staircase codes.

A. Notation

Given integers a, b where $a < b$, let $[a, b] \triangleq \{a, a+1, \dots, b\}$. For an $m \times n$ matrix \mathbf{Q} , we denote a vectorization of \mathbf{Q} by $\text{vec}(\mathbf{Q})$, where the resulting vector is assumed to be a column vector and the mapping between matrix and vector indices is given by a bijection $v : [0, m-1] \times [0, n-1] \rightarrow [0, mn-1]$. The inverse of $\text{vec}(\cdot)$ is denoted by $\text{vec}^{-1}(\cdot)$ with the underlying mapping v^{-1} , the inverse of v . For example, the mappings of the column-wise vectorization and its inverse are given by

$$v(i, j) = jm + i \quad \text{and} \quad v^{-1}(i) = (i \bmod m, \lfloor i/m \rfloor).$$

We denote the $m \times 1$ unit vector with a single 1 in the i th position by $\mathbf{e}_{m,i}$. We denote the $m \times m$ identity matrix by \mathbf{I}_m and the $m \times n$ all-zeros matrix by $\mathbf{0}_{m \times n}$. Let \mathbf{E}_m denote the $m \times m$ elementary permutation matrix, obtained by cyclically shifting each row of \mathbf{I}_m to the right by 1. Recall that for $i \geq 0$, \mathbf{E}_m^i is a permutation matrix obtained by cyclically shifting each row of \mathbf{I}_m to the right by i .

Given an $m \times n$ matrix \mathbf{A} and another matrix \mathbf{B} , their Kronecker product is defined as

$$\mathbf{A} \otimes \mathbf{B} \triangleq \begin{bmatrix} a_{00}\mathbf{B} & \dots & a_{0(n-1)}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{(m-1)0}\mathbf{B} & \dots & a_{(m-1)(n-1)}\mathbf{B} \end{bmatrix}.$$

A block diagonal matrix consisting of m copies of a matrix \mathbf{Q} along its diagonal is given by $\mathbf{I}_m \otimes \mathbf{Q}$. We denote a block diagonal matrix consisting of m arbitrary matrices $\{\mathbf{Q}_1, \mathbf{Q}_2, \dots, \mathbf{Q}_m\}$ of the same size along its diagonal by

$$\mathcal{B}(\mathbf{Q}_1, \mathbf{Q}_2, \dots, \mathbf{Q}_m) \triangleq \sum_{i=1}^m (\mathbf{e}_{m,i} \mathbf{e}_{m,i}^T) \otimes \mathbf{Q}_i.$$

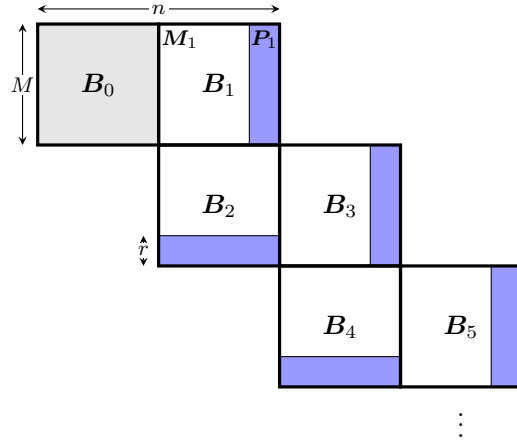


Fig. 1. Staircase code block structure. Information bits (white) and parity bits (shaded) are shown. Bits in block \mathbf{B}_0^T are fixed.

B. Encoding of staircase codes

An illustration of a staircase code is shown in Fig. 1. The fundamental building block is a binary, linear, systematic block code $C(n, k)$, referred to as a *component code*, with block-length n (required to be even) and number of information bits k . Let $R_c \triangleq k/n$ be the component code rate. For $M \triangleq n/2$, the dimension of each staircase block \mathbf{B}_i is $M \times M$. For a staircase code to have non-trivial rate (i.e., $R > 0$) the component code rate must satisfy $R_c > 1/2$.

The first staircase block \mathbf{B}_0 is fixed to all-zero bit-values. Let $r \triangleq n - k$ be the number of parity bits in a component codeword. Let \mathbf{G} be a $k \times n$ systematic generator matrix for C . We denote by \mathbf{G}_p the $k \times r$ sub-matrix of \mathbf{G} containing the r columns which correspond to the parity bits in each codeword. For $i \in \{1, 2, \dots\}$, given block $i-1$, to encode the i th block, first fill an $M \times (M-r)$ matrix \mathbf{M}_i with information bits. Next, calculate the $M \times r$ matrix \mathbf{P}_i of parity bits according to

$$\mathbf{P}_i = [\mathbf{B}_{i-1}^T \quad \mathbf{M}_i] \mathbf{G}_p \quad (1)$$

where $()^T$ denotes matrix transpose. The i th block is then given by $\mathbf{B}_i = [\mathbf{M}_i \quad \mathbf{P}_i]$.

The rate of a staircase code is given by

$$R = 2R_c - 1, \quad (2)$$

where we assumed that the smallest transmission granularity is a complete block \mathbf{B}_i .

C. Decoding of staircase codes

Staircase codes are decoded using a *sliding-window decoder*. Consider the blocks in Fig. 1 now to be received blocks buffered in a decoding window of length 6, with all except \mathbf{B}_0 corrupted by a BSC.

Decoding proceeds in iterations. Let $l \in \{1, 2, \dots, l_{\max}\}$ denote decoding iterations, with the maximum number of iterations denoted by l_{\max} . During iteration l , for each $i \in \{1, 2, \dots, 5\}$, form the matrix $[\mathbf{B}_{i-1}^T \quad \mathbf{B}_i]$ and decode each row of the matrix by a component code decoder, e.g., a syndrome decoder. Once $l = l_{\max}$ is reached, the window

“slides” by shifting out decoded block B_0 and shifting in a newly received block B_6 . The decoding process continues indefinitely in this manner.

In practice, the component code decoder can be implemented using efficient table-lookup methods for syndrome decoding to achieve very high decoding throughputs [3, Appendix] [14].

D. Motivation

Substituting $B_{i-1} = [M_{i-1} \ P_{i-1}]$ into (1), we obtain

$$P_i = [[M_{i-1} \ P_{i-1}]^T \ M_i] G_p$$

which is a linear recursion of the parity-bit matrix P_i . We refer to this as the *parity-propagation* property of staircase codes. The presence of feedback in the encoding process leads to a number of issues, the most significant of which is the lack of a termination mechanism.

Although staircase codes were designed for continuous transmission applications where termination is not necessary, allowing the encoding process to terminate after a certain number of blocks would extend their domain of application significantly. Furthermore, a terminated staircase code can be decoded by two sliding window decoders working in parallel from both ends of the code. The decoding throughput is doubled at a cost of extra hardware, a favorable trade-off in high-throughput optical-fiber systems.

III. FEED-FORWARD STAIRCASE CODE

The most pragmatic approach to mitigate the effect of parity propagation would be to not re-encode the parity bit block P_i . Such an approach is shown in Fig. 2. However, it becomes quickly obvious that this approach suffers from some important problems. Most importantly, if high-rate component codes with error correcting capability t are used, the occurrence of $t+1$ errors in the parity-part of a component code will not be corrected. Hence, if there are $t+1$ errors in the parity part of a vertical codeword, $t+1$ errors in the parity part of a horizontal codeword and an additional error in the intersection of both vertical and horizontal codewords, this additional error will not be corrected and will contribute to the error floor of the code, which will become unacceptably high for most applications. Especially in optical communications, where usually residual bit error rates in the range of 10^{-13} to 10^{-15} are required, a different approach is necessary.

In order to design a code with acceptable error floors, we adopt the parity self-protection technique proposed in [5, App. I.9] to ensure that errors in the parity part of the code do not cause large residual error floors. The structure of the proposed Feed-Forward Staircase Code (FF-SC) with parity self-protection is shown in Fig. 3. The dark shaded blocks at the bottom of even-indexed information blocks are referred to as column *redundancy blocks*. Each column redundancy block consists of a parity block \tilde{P}_c and a *self-protection* block Y . The lightly shaded blocks to the right of odd-indexed information blocks are referred to as row redundancy blocks, each consisting of a parity block \tilde{P}_r and a self-protection

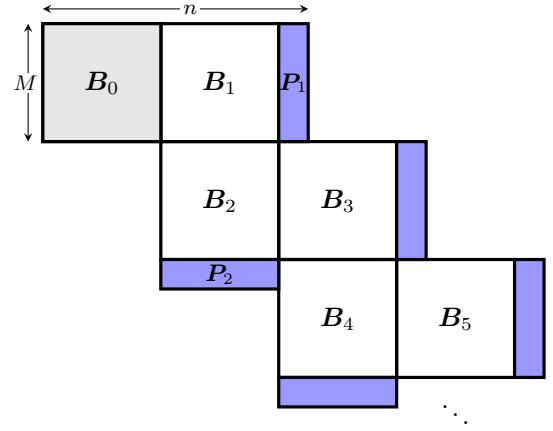


Fig. 2. Staircase codes without parity re-encoding. Information bits (white) and parity bits (shaded) are shown. Parity-bits are not used for re-encoding.

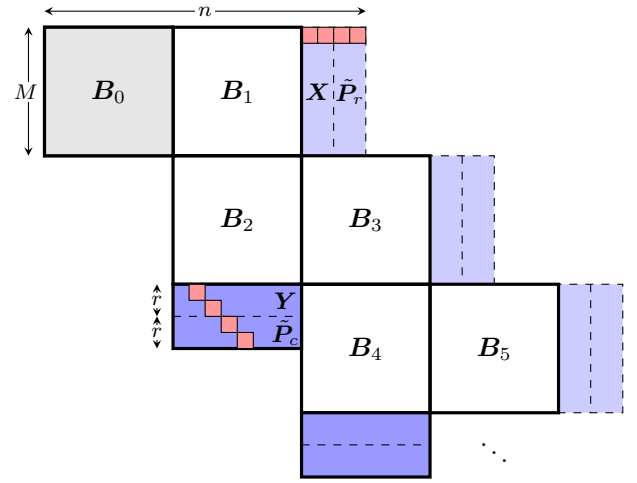


Fig. 3. Proposed feed-forward staircase code block structure. Information bits (B_i , white) and column redundancy bits (Y , \tilde{P}_c , shaded dark) are transmitted. Row redundancy bits (X , \tilde{P}_r , shaded light) are punctured. Bits in block B_0 are fixed. The small squares illustrate permutation selected for low error-floors.

block X , which are both punctured (indicated by the light shading in Fig. 3).

As in a staircase code, an FF-SC parity block contains parity bits calculated during component code encoding. The key difference in an FF-SC is that the bits in a self-protection block, which are a sub-set of the information bits of component codes, are additionally constrained.

Let π_1 and π_2 be permutations defined by

$$\pi_b(\mathbf{A}) \triangleq \text{vec}^{-1}(\mathbf{\Pi}_b \text{vec}(\mathbf{A}))$$

where $b \in \{1, 2\}$, \mathbf{A} is an $M \times r$ matrix, and $\mathbf{\Pi}_b$ is an $Mr \times Mr$ permutation matrix. By definition, π_b are bijective maps, with the property $\pi_b(\mathbf{A} + \mathbf{B}) = \pi_b(\mathbf{A}) + \pi_b(\mathbf{B})$.

We define the *self-protection constraints*

$$\mathbf{Y} = (\pi_1(\mathbf{X}))^T \quad (3)$$

$$\tilde{\mathbf{P}}_c = (\pi_2(\tilde{\mathbf{P}}_r))^T. \quad (4)$$

Since π_b are bijective, as long as the self-protection constraints are satisfied, we can puncture either the column or row

redundancy blocks. For consistency with Fig 3, we puncture the row redundancy blocks in the following.

Due to the constraints imposed on self-protection blocks, M must satisfy $2M + r = k$, hence $M = (k - r)/2$ (assuming k and r have the same parity, which can be achieved with shortening). For computing the rate, we first assume that always an even number of blocks B_i are transmitted as smallest granularity. The rate of an FF-SC is then

$$R_{\text{FF}} = 2R_c - 1 = R, \quad (5)$$

which is identical to the rate of a staircase code. If we want to achieve the finer granularity of conventional staircase codes with single blocks, we define that the parity and self-protection blocks \mathbf{Y} and $\tilde{\mathbf{P}}_c$ are attached to each block with odd index. In that case, with a total of Λ blocks transmitted we have

$$R'_{\text{FF}} = \frac{2k - n}{2k - n + 4 \lfloor \frac{\Lambda+1}{2} \rfloor \frac{1}{\Lambda} (n - k)},$$

which takes into account the potential transmission of an odd number of blocks. As $\limsup_{\Lambda \rightarrow \infty} \lfloor \frac{\Lambda+1}{2} \rfloor \frac{1}{\Lambda} = \liminf_{\Lambda \rightarrow \infty} \lfloor \frac{\Lambda+1}{2} \rfloor \frac{1}{\Lambda} = \frac{1}{2}$, we get

$$\lim_{\Lambda \rightarrow \infty} R'_{\text{FF}} = \frac{2k - n}{2k - n + 2(n - k)} = R_{\text{FF}}.$$

A. Encoding

We slightly generalize the component code definition to allow *different* binary linear block codes to be used as row and column component codes. Given block-length n and number of information bits k , let $C_r(n, k)$ be a row component code with $k \times n$ systematic generator matrix \mathbf{G} . Let $C_c(n, k)$ be a column component code with $k \times n$ systematic generator matrix \mathbf{F} . Let \mathbf{G}_p and \mathbf{F}_p denote the sub-matrices containing the r columns of \mathbf{G} and \mathbf{F} corresponding to parity-bits.

Due to the self-protection block, the last r bits out of k information bits in a component codeword are constrained. We highlight this fact by partitioning \mathbf{G}_p and \mathbf{F}_p according to

$$\mathbf{G}_p = \begin{bmatrix} \mathbf{G}_i \\ \mathbf{G}_r \end{bmatrix} \quad \mathbf{F}_p = \begin{bmatrix} \mathbf{F}_i \\ \mathbf{F}_r \end{bmatrix},$$

where \mathbf{G}_i and \mathbf{F}_i are $(k - r) \times r$ matrices and \mathbf{G}_r and \mathbf{F}_r are $r \times r$ matrices.

Consider the encoding operation over information blocks \mathbf{B}_0 , \mathbf{B}_1 , and \mathbf{B}_2 in Fig. 3. Subsequent blocks are encoded in the same manner. By horizontally concatenating \mathbf{B}_0 and \mathbf{B}_1 , we obtain

$$\mathbf{P}_r = [\mathbf{B}_0 \quad \mathbf{B}_1] \mathbf{G}_i.$$

By vertically concatenating \mathbf{B}_1 and \mathbf{B}_2 , we obtain

$$\mathbf{P}_c = \mathbf{F}_i^T \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \end{bmatrix}.$$

Note that \mathbf{P}_r and \mathbf{P}_c are not the same as $\tilde{\mathbf{P}}_r$ and $\tilde{\mathbf{P}}_c$.

Consider the entries of the $M \times r$ matrix \mathbf{X} and the $r \times M$ matrix \mathbf{Y} to be variables. According to the structure shown in Fig. 3, we can write $\tilde{\mathbf{P}}_r$ and $\tilde{\mathbf{P}}_c$ as

$$\tilde{\mathbf{P}}_r = \mathbf{P}_r + \mathbf{X} \mathbf{G}_r, \quad \tilde{\mathbf{P}}_c = \mathbf{P}_c + \mathbf{F}_r^T \mathbf{Y}.$$

Imposing self-protection conditions (3) and (4), we obtain

$$\mathbf{P}_c + (\pi_2(\mathbf{P}_r))^T = \mathbf{F}_r^T \mathbf{Y} + (\pi_2(\pi_1^{-1}(\mathbf{Y}^T) \mathbf{G}_r))^T.$$

Each of the above terms is an $r \times M$ matrix. Let $\text{vec}(\cdot)$ be the column-wise vectorization and let $\mathbf{y} = \text{vec}(\mathbf{Y})$, $\mathbf{p}_c = \text{vec}(\mathbf{P}_c)$, and $\mathbf{p}_r = \text{vec}((\pi_2(\mathbf{P}_r))^T)$. Let $\mathbf{\Pi}_T$ be the permutation matrix satisfying $\mathbf{Y}^T = \text{vec}^{-1}(\mathbf{\Pi}_T \text{vec}(\mathbf{Y}))$. Using the fact that for some matrix \mathbf{Q}

$$\text{vec}(\mathbf{Q} \mathbf{Y}) = (\mathbf{I}_M \otimes \mathbf{Q}) \text{vec}(\mathbf{Y}),$$

the above expression can be written as

$$\mathbf{p}_c + \mathbf{p}_r = [\mathbf{I}_M \otimes \mathbf{F}_r^T + \mathbf{\Pi}_T \mathbf{\Pi}_2 \mathbf{\Pi}_T (\mathbf{I}_M \otimes \mathbf{G}_r^T) \mathbf{\Pi}_T \mathbf{\Pi}_1^{-1} \mathbf{\Pi}_T] \mathbf{y} \triangleq \mathbf{A} \mathbf{y}.$$

If \mathbf{A} is invertible, then the matrix \mathbf{Y} is given by

$$\begin{aligned} \mathbf{y} &= \mathbf{A}^{-1}(\mathbf{p}_c + \mathbf{p}_r) \\ &\triangleq \mathbf{A}^{-1} \mathbf{c}. \end{aligned} \quad (6)$$

The invertibility of \mathbf{A} depends on the choices of $C_r(n, k)$, $C_c(n, k)$, \mathbf{F} , \mathbf{G} , $\mathbf{\Pi}_1$, and $\mathbf{\Pi}_2$. Using the same row and column component codes, we have found that searching over the space of all $\mathbf{\Pi}_1$ and $\mathbf{\Pi}_2$ can quickly produce an invertible \mathbf{A} . The search and calculation of \mathbf{A}^{-1} can be performed offline at design time, since information bits are only involved in the calculation of \mathbf{c} .

The main complexity of FF-SC encoding is the multiplication in (6) between an $Mr \times Mr$ matrix and an $Mr \times 1$ vector. The complexity of this operation highly depends on the choice of permutation matrices $\mathbf{\Pi}_1$ and $\mathbf{\Pi}_2$. For instance, the permutation matrices may be chosen such that the hardware implementation is simplified or such that \mathbf{A}^{-1} has a special structure easing the multiplication.

B. Decoding

Decoding of FF-SC is very similar to conventional staircase codes. A sliding window decoder is used starting from block \mathbf{B}_0 . When corrections are made in a column redundancy block the corresponding row redundancy block is also modified, and vice versa. Additional logic is required to implement the permutations π_1 , π_2 , and their inverses.

C. Low error-floor permutations

We describe a choice of permutations π_1 and π_2 suitable for applications requiring very low error-floors. The permutations π_1 , π_2 are defined by the permutation matrices

$$\begin{aligned} \mathbf{\Pi}_1 &= \mathcal{B}(\mathbf{E}_M^{M-1}, \mathbf{E}_M^{M-2}, \dots, \mathbf{E}_M^{M-r}) \\ \mathbf{\Pi}_2 &= \mathcal{B}(\mathbf{E}_M^{M-r-1}, \mathbf{E}_M^{M-r-2}, \dots, \mathbf{E}_M^{M-2r}), \end{aligned}$$

together with column-wise vectorization $\text{vec}(\cdot)$ and its inverse $\text{vec}^{-1}(\cdot)$.

These permutations cyclically shift each column of \mathbf{X} and $\tilde{\mathbf{P}}_r$ by a number of bits related to their column index, an example of which is shown in Fig. 3. They can be implemented efficiently in hardware using barrel shifters. Discussions of the estimated and simulated error-floor performance under these permutations are given in Sec. V.

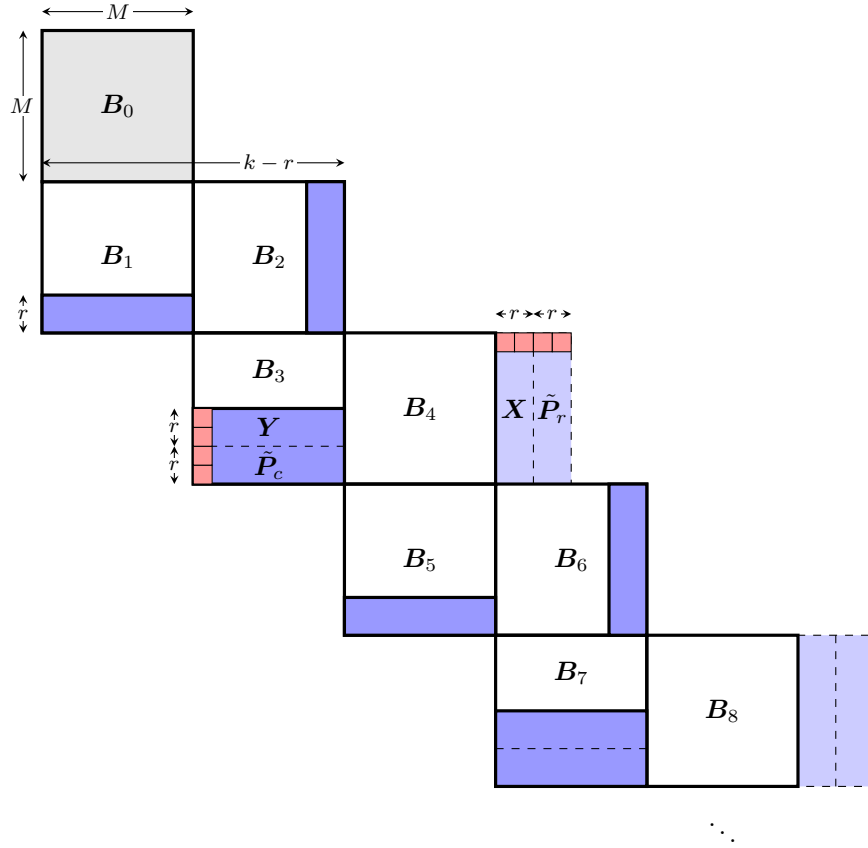


Fig. 4. Partial feed-forward staircase code block structure. Information bits (B_i , white), parity bits (shaded dark), and column-redundancy bits (Y, \tilde{P}_c , shaded dark) are transmitted. Row redundancy bits (X, \tilde{P}_r , shaded light) are punctured. Bits in block B_0 are fixed. The small squares illustrate the trivial permutations.

IV. PARTIAL FEED-FORWARD STAIRCASE CODE

Although self-protection allows us to considerably reduce the error floor of feed-forward staircase codes, the error floor (see Sec. VI) may still be unacceptably high for some applications requiring very low residual BERs, e.g., optical core networks. We therefore slightly relax the parity-propagation constraint by allowing the parity bits to propagate over some blocks and introduce *Partial Feed-Forward Staircase Codes* (PFF-SCs).

Let $L \in \{1, 2, \dots\}$ be the *propagation length* of a PFF-SC, defined as the maximum number of consecutive blocks over which parity-propagation can occur. The PFF-SC then uses a hybrid structure, with $L - 1$ blocks being standard staircase code blocks followed by one block with parity bits that are not re-encoded but where self-protection is used to mitigate the detrimental effect of harmful error patterns. The self-protection scheme also results in one block containing only information bits. Figure 4 illustrates the structure of a PFF-SC with $L = 3$. In this example, 2 out of every 4 blocks are standard staircase code blocks and 1 out of every 4 blocks contains only information bits. Self-protection is used to stop parity-propagation after $L = 3$ blocks.

Another major difference in PFF-SCs is the position of the self-protection redundancy blocks, which are part of the conventional staircase structure. This modification allows the permutations π_1, π_2 to be trivial and drastically reduces the

error-floor as compared to FF-SC (see Sec. VI). Another difference is that the number of information bits per block B_i is not constant. As in FF-SC, we set $M = (k - r)/2$ to account for the self-protection and all blocks contain M^2 code bits. The component codes are shortened respectively. In order to accommodate the position of self-protection redundancy blocks Y , the component codes involved in self-protection (e.g., codes over blocks B_2 and B_3 as well as B_6 and B_7 in Fig. 4) must be shortened by an extra $2r$ bits relative to the other component codes.

A. Rate of PFF-SCs

In order to compute the rate of PFF-SCs, we count the number of information bits per block. The first $L - 1$ blocks $B_{1+(L+1)i}, \dots, B_{L-1+(L+1)i}$, $i \in \{0, 1, \dots\}$ out of $L + 1$ blocks (e.g., B_1 and B_2 in Fig. 4) are standard staircase code blocks of size $M \times M$ with $M(M - r) = \frac{1}{4}(k^2 + 3r^2 - 4kr)$ information bits. The block $B_{L+(L+1)i}$, $i \in \{0, 1, \dots\}$ contains exactly $M(M - 2r) = \frac{1}{4}(k^2 + 5r^2 - 6kr)$ information bits and finally, the block $B_{(L+1)(i+1)}$, $i \in \{0, 1, \dots\}$ contains exactly $M^2 = \frac{1}{4}(k - r)^2$ information bits. For computing the rate, we must fix again the granularity of transmission. If we assume that always $L + 1$ blocks $B_1, \dots, B_{(L+1)i}$, $i \in \mathbb{N}$ are

transmitted, then the rate can be computed as

$$\begin{aligned} R_{\text{PFF}} &= \frac{(L-1)M(M-r) + M(M-2r) + M^2}{(L+1)M^2} \\ &= 1 - \frac{r}{M} = 1 + \frac{2R_c - 2}{2R_c - 1}, \end{aligned} \quad (7)$$

which is independent of L . As $R_{\text{PFF}} - R = \frac{(2R_c - 2)^2}{1 - 2R_c}$, we can conclude that $R_{\text{PFF}} < R$ as $R_c > \frac{1}{2}$ has to hold (see Sec. II-B). However, at high rates the differences are small. For example, R_{PFF} is within 5% of R for $R_c \geq 10/11$ and within 25% for $R_c \geq 5/6$. Note that a PFF-SC of non-trivial rate requires $R_c > 3/4$.

This result may seem counter-intuitive at first glance, since it appears that we should recover the original staircase code rate R for $L \rightarrow \infty$. However, contrary to the original staircase code construction (see Sec. II), in the proposed construction the component codes of the staircase-like blocks are shortened by $2r$, which leads to the observed rate difference. We could relax the granularity constraint of $L+1$ blocks and find an expression for $R'_{\text{PFF}}(\Lambda, L)$. As this expression is cumbersome and does not lead to any new insights, we omit it here. For practical purposes, it is customary to restrict ourselves to the granularity of $L+1$ blocks, allowing for easy termination and avoiding possibly higher error rates at the code boundaries.

B. Description of the Encoder

In this subsection, we describe the encoder of PFF-SC. We focus only on the self-protection blocks, since $L-1$ out of $L+1$ consecutive blocks are encoded in the same way as the original staircase code. Our explanations will focus on Fig. 5, which highlights blocks B_2 , B_3 , B_4 , Y , \tilde{P}_c , X , and \tilde{P}_r of Fig. 4 for $L=3$.

Figure 5 further sub-divides each block into sub-blocks. The encoding process consists of two stages. Stage 1 calculates Y_1 . Stage 2 calculates Y_2 based on Y_1 . In terms of implementation complexity, stage 1 is equivalent to component code encoding while stage 2 is a general matrix multiplication. Fortunately, for high code rates where $M \gg r$, the encoding complexity is dominated by stage 1.

1) *Calculating Y_1* : We inherit the definitions of matrices G_p , F_p , G_i , F_i , and G_r , F_r from Sec. III. By horizontally concatenating $M_{1,1}$, $M_{1,2}$, and $M_{2,1}$, we obtain

$$P_{r,1} = [M_{1,1} \quad M_{1,2} \quad M_{2,1}] G_i. \quad (8)$$

By vertically concatenating blocks $0_{2r \times M-2r}$, $M_{0,1}$ and $M_{1,1}$, where $0_{2r \times M-2r}$ accounts for the extra shortening of the column component codes, we obtain

$$P_{c,1} = F_i^T \begin{bmatrix} 0_{2r \times M-2r} \\ M_{0,1} \\ M_{1,1} \end{bmatrix}.$$

We write $\tilde{P}_{c,1}$ and $\tilde{P}_{r,1}$ as

$$\tilde{P}_{c,1} = P_{c,1} + F_r^T Y_1, \quad \tilde{P}_{r,1} = P_{r,1} + X_1 G_r.$$

Imposing self-protection constraints

$$Y_1 = X_1^T, \quad \tilde{P}_{c,1} = \tilde{P}_{r,1}^T$$

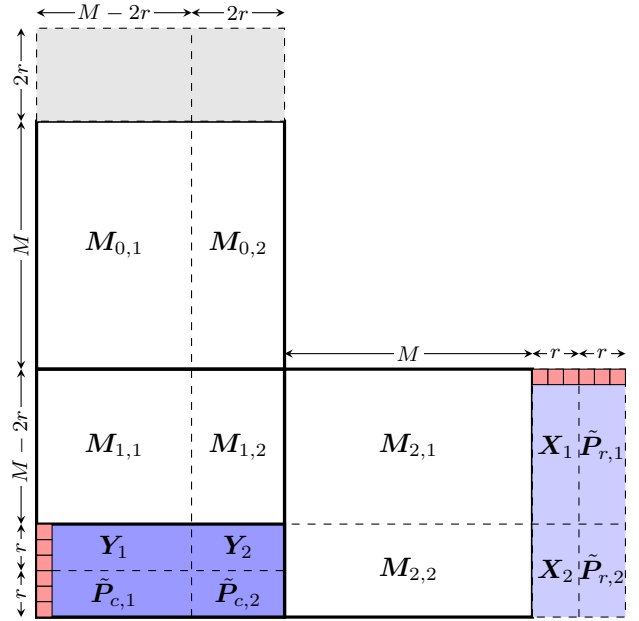


Fig. 5. Sub-block divisions for PFF-SC encoding. The $2r \times M$ sub-block at the top (shaded light) is shortened.

under trivial permutations and solving for Y_1 gives

$$\begin{aligned} Y_1 &= (G_r^T + F_r^T)^{-1} (P_{c,1} + P_{r,1}^T) \\ &\triangleq A^{-1} (P_{c,1} + P_{r,1}^T). \end{aligned} \quad (9)$$

Since $A = 0_{r \times r}$ if $G_r = F_r$, a necessary condition for A to be invertible is $G_r \neq F_r$. Here we satisfy this condition by using different binary cyclic codes as row and column component codes. However, instead of using different component codes with different error correction capabilities and potentially requiring distinct decoder hardware implementations, we propose the following construction: Let $g(x)$ and $f(x)$ be generator polynomials for $C_r(n, k)$ and $C_c(n, k)$. We require $g(x)$ and $f(x)$ to satisfy the condition

$$f(x) = x^{\deg(g(x))} g(x^{-1}) \quad (10)$$

where $\deg(p(x))$ is the degree of the polynomial $p(x)$. The component codes then have the property that the “mirror-image” of a codeword $(c_0, c_1, \dots, c_{n-1}) \in C_r(n, k)$, i.e., $(c_{n-1}, c_{n-2}, \dots, c_0)$, is a codeword of $C_c(n, k)$, and vice versa [15, Ch. 7]. Hence, the same decoder hardware can be reused to decode both component codes, with some simple bit-reversal logic.

Using different binary cyclic component codes with generator polynomials satisfying (10) gives an invertible A as $G_r \neq F_r$. By calculating A^{-1} offline at design time, the complexity of finding Y_1 and $\tilde{P}_{c,1}$ at encoding time is equivalent to a multiplication between an $r \times r$ matrix and an $r \times M-2r$ matrix.

2) *Calculating Y_2* : In stage 2, the blocks Y_1 and $\tilde{P}_{c,1}$ are considered known. By vertically concatenating blocks $0_{2r \times 2r}$, $M_{0,2}$ and $M_{1,2}$ we obtain

$$P_{c,2} = (F_i^T)^T \begin{bmatrix} 0_{2r \times 2r} \\ M_{0,2} \\ M_{1,2} \end{bmatrix}$$

hence

$$\tilde{P}_{c,2} = P_{c,2} + F_r^T Y_2. \quad (11)$$

We partition the matrix G_i into 3 sub-matrices with

$$G_i = \begin{bmatrix} G_A \\ G_B \\ G_C \end{bmatrix}$$

where $\dim G_A = (M - 2r) \times r$, $\dim G_B = 2r \times r$, and $\dim G_C = M \times r$. We can now write

$$\tilde{P}_{r,2} = \begin{bmatrix} Y_1 \\ \tilde{P}_{c,1} \end{bmatrix} G_A + \begin{bmatrix} Y_2 \\ \tilde{P}_{c,2} \end{bmatrix} G_B + M_{2,2} G_C + X_2 G_r.$$

Using (11) and the self-protection constraint $Y_2^T = X_2$, we have

$$\tilde{P}_{r,2} = \begin{bmatrix} Y_1 & 0_{r \times 2r} & M_{2,2} \\ \tilde{P}_{c,1} & P_{c,2} & \end{bmatrix} G_i + \begin{bmatrix} I_r \\ F_r^T \end{bmatrix} Y_2 G_B + Y_2^T G_r.$$

Imposing the self-protection constraint $\tilde{P}_{r,2} = (\tilde{P}_{c,2})^T$ and simplification yields

$$Y_2^T A^T + \begin{bmatrix} I_r \\ F_r^T \end{bmatrix} Y_2 G_B = C \quad (12)$$

where A was defined implicitly in (9) and with

$$C \triangleq \begin{bmatrix} Y_1 & 0_{r \times 2r} & M_{2,2} \\ \tilde{P}_{c,1} & P_{c,2} & \end{bmatrix} G_i + [0_{2r \times 2r} \quad M_{0,2}^T \quad M_{1,2}^T] F_i.$$

Note that all terms in (12) are $2r \times r$ matrices.

Let $\text{vec}(\cdot)$ now denote the *row-wise* vectorization given by the mapping $v(i, j) = in + j$. Let $y = \text{vec}(Y_2)$ and $c = \text{vec}(C)$. Let $S(A)$ be the $r \times 2r^2$ matrix where for $i \in [0, r-1]$ and $j = 2ri$, the j th column of $S(A)$ is the i th column of A , with zeros elsewhere. We can then equivalently write (12) as

$$By = c$$

where B is the $2r^2 \times 2r^2$ matrix given by

$$B \triangleq \begin{bmatrix} S(A) \\ S(A)E_{2r^2} \\ \vdots \\ S(A)E_{2r^{2-1}} \end{bmatrix} + \begin{bmatrix} I_r \otimes G_B^T \\ F_r^T \otimes G_B^T \end{bmatrix}.$$

3) *Finding an invertible B* : Since G_r and F_r were fixed in stage 1 in order to obtain an invertible A , if B is singular, the only way to obtain an invertible B is to manipulate G_B using elementary row operations. Here we focus on row permutations of G_B only, since they do not affect the error floor.

Let Π be a $2r \times 2r$ permutation matrix. Denote the permuted G_B by $\tilde{G}_B \triangleq \Pi G_B$. A computer search can be used to find an appropriate Π that results in an invertible B .

Given Π , the expressions for $\tilde{P}_{r,2}$ and B are modified by replacing G_B with \tilde{G}_B . Note that Π also affects stage 1 calculations, where (8) has to be modified to

$$P_{r,1} = [M_{1,1} \quad M_{1,2}\Pi \quad M_{2,1}] G_i.$$

For an invertible B , the matrix Y_2 is given by

$$y = B^{-1}c.$$

The complexity of calculating Y_2 is dominated by the multiplication with a $2r^2 \times 2r^2$ matrix. Since only 1 out of every $L + 1$ blocks requires self-protection calculations, the average complexity of PFF-SC approaches conventional staircase codes with increasing L .

V. ERROR-FLOOR ANALYSIS

Error-floor analysis of staircase codes and its variants proposed in this paper depends on enumerating the number of *stall patterns*, i.e., patterns of errors that the decoder cannot remove [3], [16]. To obtain a simple estimate of the error-floor, we only enumerate the smallest stall patterns resulting from channel errors, referred to as *minimal* stall patterns.

We consider an erroneously decoded bit to be a bit error only if it is an information bit. A decoded block is considered to be a block error if it contains at least one bit error. The block (BKER) and bit (BER) error-rates are defined according to these definitions.

We estimate the block and bit error-floors of FF-SC based on low-error-floor permutations of Sec. III-C assuming transmission over a BSC with error probability p . An example of a minimal stall pattern for component codes with $t = 3$ is shown in Fig. 6, consisting of 4 information-bit errors and 4 redundancy-bit errors from the channel.

To construct such a stall pattern, first choose any 2 out of M rows in the information block, such as the rows marked by the horizontal dashed and dash-dotted lines in Fig. 6. Denote the chosen rows by r_1 and r_2 . Under the transposes in (3) and (4), the chosen rows are mapped to columns marked by the *thin* vertical dashed and dash-dotted lines, reflected about the diagonal of the information block.

Under the proposed low-error-floor permutations, bit errors in the row redundancy block are cyclically shifted by no more than $2r - 1$ columns, modulo M , in the column redundancy block. In Fig. 6, the range of cyclic shifts is bounded by the thin and corresponding *thick* vertical lines. For example, bit errors in the row redundancy block of r_1 may be shifted to columns within the thin and thick dashed vertical lines. For r_2 , bit errors in the row redundancy block may be shifted to columns within the thin and thick dash-dotted vertical lines, wrapping around the right boundary of the column redundancy block. Given r_i , we define its *valid column set* by

$$S(r_i) \triangleq \{r_i + j \bmod M \text{ for all } j \in [0, 2r - 1]\}.$$

It is simple to verify the following *spreading property* of the low-error-floor permutations: if $2r < M$, i.e., $R > 1/2$ or $OH < 100\%$ (where OH denotes the overhead of the code, defined as $OH \triangleq (1/R - 1) \times 100\%$), then row redundancy block bit-errors belonging to the same row *cannot* belong to the same column in the column redundancy block. Consequently, columns in the stall pattern can only be chosen from the *intersection* of valid column sets. The number of choices of such columns is

$$|S(r_1) \cap S(r_2)| \leq 2r. \quad (13)$$

In Fig. 6, the intersection consists of columns bounded between the thin dashed and thick dash-dotted vertical lines

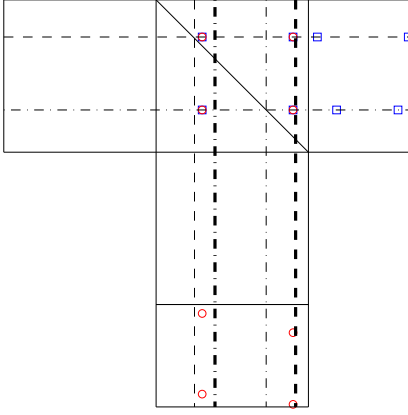


Fig. 6. Minimal stall pattern used to estimate FF-SC error floors for component codes with $t = 3$. Blue (\square) markers are bit errors in row component codes. Red (\circ) markers are bit errors in column component codes. Dashed and dash-dotted lines are referred to in the derivation of error-floor estimates in Sec. V. Note that only 4 out of the 8 bit errors in redundancy blocks are received from the channel, the other ones are interleaved versions thereof.

and columns bounded between the thin dash-dotted and thick dashed vertical lines. The resulting error-floor estimates based on the simple upper-bound (13) are given by

$$\text{BKER}_{\text{FF}} \approx \binom{M}{2} \binom{2r}{2} p^8$$

$$\text{BER}_{\text{FF}} \approx \text{BKER}_{\text{FF}} \frac{4}{M^2}.$$

where p denotes the error probability of the BSC.

For arbitrary t , let $t_i = \lfloor (t+1)/2 \rfloor$ and $t_r = t+1-t_i$. For odd t , $t_i = t_r$ and the above argument for $t = 3$ applies directly. Observe that t_i (resp. t_r) is then the number of information (resp. redundancy) block bit-errors in each row of a minimal stall pattern. The error-floor estimates for odd t are given by

$$\text{BKER}_{\text{FF}} \approx \binom{M}{t_r} \binom{2r}{t_r} p^{t_r(t+1)} \quad (14)$$

$$\text{BER}_{\text{FF}} \approx \text{BKER}_{\text{FF}} \frac{t_i t_r}{M^2}. \quad (15)$$

For even t , we first choose t_i rows out of M in the information block. Each erroneous row is assumed to contain t_i bit errors in the information block and t_r bit errors in the row redundancy block. Under the spreading property, bit errors in the row redundancy block are spread to at least t_r distinct columns in the column redundancy block. In the minimal stall pattern, there are *exactly* t_r erroneous columns in the column redundancy block, each containing t_i bit-errors (since the total number of bit errors in the row redundancy block is $t_i t_r$). Consequently, there must be t_r erroneous columns in the information block, each containing at least $t+1-t_i = t_r$ bit-errors. We add one additional erroneous row, with t_i bit errors in the information block and t_r bit errors in the row redundancy block, to complete the minimal stall pattern.

The resulting minimal stall pattern contains $t_i t_r$ bit errors in the information block and t_r^2 bit-errors in the row (or column) redundancy block for a total of $t_r(t_i+t_r) = t_r(t+1)$ bit errors.

TABLE I
FEED-FORWARD STAIRCASE CODE PARAMETERS

R	$OH(\%)$	m	t	s	M
3/4	33.3	8	3	63	72
4/5	25.0	8	3	15	96
5/6	20.0	9	3	187	135
13/14	7.69	10	3	183	390

Applying the intersection of valid column sets argument for the number of choices of columns in the stall pattern, we conclude that the error-floor estimates for even t are also given by (14) and (15).

We estimate the block and bit error-floors of PFF-SC based on the minimal stall pattern of weight 16, with all 16 bits being information bits. This is the same minimal stall pattern as in the original staircase codes [3], obtained by choosing $t+1$ rows out of M followed by k columns out of M in one block and $t+1-k$ columns out of M in the adjacent block, for all $k \in [0, 3]$. The error-floor estimates for general t are given by

$$\text{BKER}_{\text{PFF}} \approx \binom{M}{t+1} \sum_{k=0}^t \binom{M}{k} \binom{M}{t+1-k} p^{(t+1)^2}$$

$$\text{BER}_{\text{PFF}} \approx \text{BKER}_{\text{PFF}} \frac{(t+1)^2}{M^2}.$$

VI. SIMULATION EXAMPLE

In this section, we consider FF-SC and PFF-SC based on shortened primitive BCH component codes. Let $m > 0$ be the *degree of the extension field* and $t > 0$ be the *unique decoding radius* of a primitive BCH code. Let $s \geq 0$ be the number of bits to shorten each BCH code in order to obtain a component code with block-length n and number of information bits k . Given n and k , the values of m , t , and s are determined by the constraints

$$n = 2^m - 1 - s, \quad k = n - mt.$$

For fixed t , we always choose the smallest m that satisfies these constraints.

Given t and the primitive element $\alpha \in \text{GF}(2^m)$, the row generator polynomial is given by $g(x) = \prod_{i \in [1, 2t]} M_{\alpha^i}(x)$ where $M_{\alpha^i}(x)$ is the minimal polynomial of α^i . The column generator polynomial, which we choose to be the reciprocal polynomial of $g(x)$, is given by $f(x) = \prod_{i \in [1, 2t]} M_{\alpha^{-i}}(x)$ where

$$M_{\alpha^{-i}}(x) \triangleq x^{\deg(M_{\alpha^i}(x))} M_{\alpha^i}(x^{-1}).$$

We constructed FF-SC and PFF-SC of rates $R \in \{3/4, 4/5, 5/6, 13/14\}$. The code parameters are shown in Tables I and II. We chose $t = 3$ so that error-floors can be studied in the simulation. Furthermore, the selection of $t = 3$ yields a very efficient decoder based on lookup tables [3].

Software simulated block and bit error-probabilities of transmission over a BSC using the codes of Tables I and II are shown in Fig. 7, along with their error-floor estimates (shown as thin lines with open markers). All FF-SCs were implemented using the low-error-floor permutations of Sec. III-C. All PFF-SCs were implemented with $L = 1$.

TABLE II
PARTIAL FEED-FORWARD STAIRCASE CODE PARAMETERS

R	OH (%)	m	t	s	M	p_{15}	Δ	Δ_{ref}
3/4	33.3	8	3	15	96	$1.82 \cdot 10^{-2}$	1.64	1.38
4/5	25.0	9	3	187	135	$1.56 \cdot 10^{-2}$	1.25	1.06
5/6	20.0	9	3	133	162	$1.30 \cdot 10^{-2}$	1.07	0.92
13/14	7.69	10	3	123	420	$4.80 \cdot 10^{-3}$	0.73	0.48

Both proposed classes of codes show similar performance in the waterfall region. PFF-SCs have a slight performance loss at lower rates due to their rate loss, which requires a larger M compared to an FF-SC of the same rate.

In the error-floor region, even with low-error-floor permutations, FF-SCs have observable error-floors. On the other hand, PFF-SCs, due to their similarity to the structure of the original staircase codes, do not exhibit any bit error-floor above a BER of 10^{-15} . In fact, the estimates of the bit error-floor are orders of magnitude below 10^{-15} . For comparison, we also give the bit error rates of the original staircase codes (\blacklozenge) constructed using the same component codes. We can see that the original staircase code slightly outperforms the FF-SC and PFF-SC, especially for low rates, however, at high rates, the difference becomes negligible. This difference is most likely due to the stronger coupling between blocks in the original staircase code construction.

Let $h(x)$ be the binary entropy function and $\text{erfc}^{-1}(x)$ be the inverse complementary error function. Given a code of rate R which achieves an output BER of 10^{-15} at an input BER of p_{15} , we define the NCG gap to capacity (in dB) by

$$\Delta \triangleq 20 \log_{10} \text{erfc}^{-1}(2h^{-1}(1-R)) - 20 \log_{10} \text{erfc}^{-1}(2p_{15})$$

where $h^{-1}(x)$ is the unique $0 \leq p < 1/2$ such that $h(p) = x$.

We extrapolate the BER curves of PFF-SC down to 10^{-15} in order to estimate p_{15} . The values of p_{15} with the corresponding Δ are given in Table II. For comparison, we also included the Δ_{ref} of staircase codes of the same rates from [4], which were found by exhaustively searching over a wide range of parameters m and t and are considered to be the best staircase codes based on the construction given in Sec. II and [3]. The referenced codes were based on BCH component codes with $t \in \{4, 5\}$. Nevertheless, the difference in NCG between PFF-SCs with $t = 3$ and the reference codes are less than 0.26 dB. Error-floors of PFF-SCs and the reference codes are identical and well below 10^{-15} .

VII. CONCLUSIONS

In this paper, we proposed two modifications to staircase codes which allow for convenient termination. In feed-forward staircase codes, a self-protection technique is used to completely eliminate parity-propagation. In partial feed-forward staircase codes, a propagation-length parameter is used to control the extent of parity-propagation.

Analysis and simulation results show that these codes have similar performance as the original staircase codes. FF-SC have slightly better waterfall performance than PFF-SC, while PFF-SC have much lower error-floors. Hence, FF-SC and PFF-SC are good staircase code solutions for applications where parity-propagation is undesirable or termination is necessary.

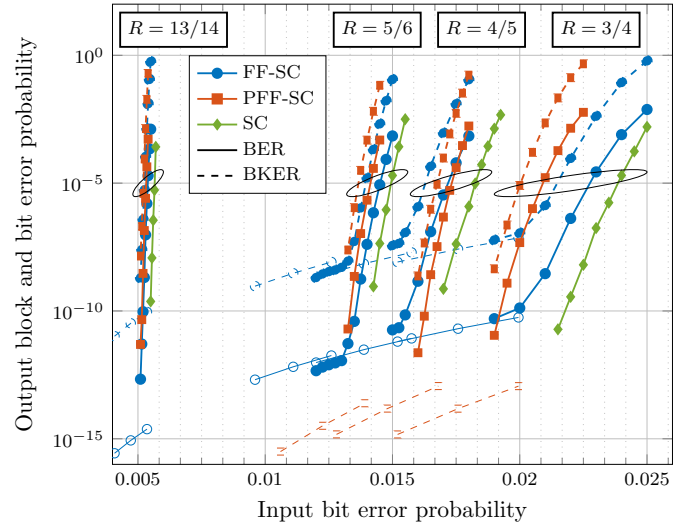


Fig. 7. Block (dashed lines) and bit-error probabilities (solid lines) of feed-forward (\bullet) and partial feed-forward (\blacksquare) staircase codes with parameters in Tables I and II. For reference, conventional staircase codes (\blacklozenge) are also shown. Block and bit error-floor estimates are also shown (thin lines, open markers).

REFERENCES

- [1] A. Leven and L. Schmalen, "Status and recent advances on forward error correction technologies for lightwave systems," *J. Lightw. Technol.*, vol. 32, no. 16, pp. 2735–2750, Aug. 2014.
- [2] B. S. G. Pillai, B. Sedighi, K. Guan, N. P. Anthapadmanabhan, W. Shieh, K. J. Hinton, and R. S. Tucker, "End-to-end energy modeling and analysis of long-haul coherent transmission systems," *J. Lightw. Technol.*, vol. 32, no. 18, pp. 3093–3111, Sep. 2014.
- [3] B. P. Smith, A. Farhood, A. Hunt, F. R. Kschischang, and J. Lodge, "Staircase codes: FEC for 100Gb/s OTN," *J. Lightw. Technol.*, vol. 30, no. 1, pp. 110–117, Jan. 2012.
- [4] L. M. Zhang and F. R. Kschischang, "Staircase codes with 6% to 33% overhead," *J. Lightw. Technol.*, vol. 32, no. 10, pp. 1999–2002, May 2014.
- [5] "ITU-T recommendation G.975.1, Forward error correction for high bit-rate DWDM submarine systems," Feb. 2004, Series G: Transmission systems and media, digital systems and networks. International Telecommunication Union.
- [6] C. Häger, A. Graell i Amat, H. Pfister, A. Alvarado, Brännström, and E. Agrell, "On parameter optimization for staircase codes," in *Proceedings of Optical Fiber Communication Conference and Exposition (OFC)*, Los Angeles, CA, USA, Mar. 2015.
- [7] Y.-Y. Jian, H. D. Pfister, K. R. Narayanan, R. Rao, and R. Mazareh, "Iterative hard-decision decoding of braided BCH codes for high-speed optical communication," in *Proceedings of Global Communications Conference (GLOBECOM 2013)*, Atlanta, GA, USA, Dec. 2013, pp. 2398–2403.
- [8] Y.-Y. Jian, H. Pfister, and K. Narayanan, "Approaching capacity at high rates with iterative hard-decision decoding," in *Proc. IEEE ISIT*, Jul. 2012, pp. 2696–2700.
- [9] Y.-Y. Jian, "On the analysis of spatially-coupled GLDPC codes and the weighted min-sum algorithm," Ph.D. dissertation, Texas A&M University, Aug. 2013.
- [10] C. Häger, H. D. Pfister, A. Graell i Amat, and F. Brännström, "Density evolution for deterministic generalized product codes on the binary erasure channel," *arXiv preprint arXiv:1512.00433*, 2015.
- [11] L. M. Zhang, D. Truhachev, and F. Kschischang, "Spatially-coupled split-component codes with iterative algebraic decoding," *arXiv preprint arXiv:1512.01132*, 2015.
- [12] M. Tavares, "On low-density parity-check convolutional codes: Constructions, analysis and VLSI implementations," Ph.D. dissertation, TU Dresden, Dresden, Germany, 2010.
- [13] J. Wu, "A survey of WDM network reconfiguration: Strategies and triggering methods," *Computer Networks*, vol. 55, no. 11, pp. 2622–2645, Aug. 2011.

- [14] R. Chien, B. Cunningham, and I. Oldham, "Hybrid methods for finding roots of a polynomial with application to BCH decoding," *IEEE Trans. Inf. Theory*, vol. 15, no. 2, pp. 329–335, Mar. 1969.
- [15] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, Netherlands: North Holland, 1977.
- [16] J. Justesen, "Performance of product codes and related structures with iterated decoding," *IEEE Trans. Commun.*, vol. 59, no. 2, pp. 407–415, Feb. 2011.